Area Convergence of Monoculus Robots with Additional Capabilities

Debasish Pattanayak[1], Kaushik Mondal[2], Partha Sarathi Mandal[1] and Stefan Schmid[3]

[1]Department of Mathematics, Indian Institute of Technology Guwahati, India
[2]Department of Mathematics, Indian Institute of Technology Ropar, India
[3]Faculty of Computer Science, University of Vienna, Austria
Email: psm@iitg.ac.in

This paper considers the area convergence problem, that requires a group of robots to gather in a small area not defined *a priori*. While it is known that robots can gather at a point if they can precisely measure distances, we, in this paper, show that without any agreement on the coordinate system, it is impossible for robots to converge to an area if they cannot measure distances or angles. We denote these robots without the ability to measure distances or angles as *monoculus robots*. We present a counterexample showing that monoculus robots fail in area convergence even with the capability of measuring angles. However, monoculus robots with a weak notion of distance or minimal agreement on the coordinate system are sufficient to achieve area convergence. In particular, we present area convergence algorithms in asynchronous model for such monoculus robots with one of the two following simple additional capabilities: (1) Locality Detection ($\mathcal{LD}$), a notion of distance *or* (2) Orthogonal Line Agreement ($\mathcal{OLA}$), a notion of direction. We discuss extensions corresponding to multiple dimensions and the termination. Additionally, we validate our findings using simulation and show the robustness of our algorithms in the presence of errors in observation or movement.

Keywords: Area Convergence; Monoculus Robots; Oblivious Mobile Robots; Asynchronous; Distributed Algorithm

Received 00 January 2009; revised 00 Month 2009

1. INTRODUCTION

1.1. The Context: Tiny Robots

In recent years, there has been a wide interest in the cooperative behavior of tiny robots. In particular, many distributed coordination protocols have been devised for a wide range of models and for a wide range of problems, like convergence, gathering, pattern formation, flocking, etc. [1–4]. At the same time, researchers have also started characterizing the scenarios in which such problems cannot be solved, deriving impossibility results [5, 6].

1.2. Our Motivation: Even Simpler Robots

An interesting question regards the minimal cognitive capabilities that such tiny robots need to have for completing a specific task. In particular, researchers have initiated the study of "weak robots" [7]. Weak robots are *anonymous* (they do not have any identifier), *autonomous* (they work independently), *homogeneous* (they behave the same in the same situation), and *silent* (they also do not communicate with each other). Weak robots are usually assumed to have their own local view, represented as a Cartesian coordinate system with an origin, unit length, and axes. The robots may not agree on the orientation of axes, or the *chirality* (relative order of the orientation of axes or handedness). The robots move in a sequence of three consecutive actions *Look-Compute-Move*: they observe the positions of other robots in their local coordinate system, and the observation step returns a set of points to the observing robot. The robots cannot distinguish if there are multiple robots at the same position, i.e., they do not have the capability of *multiplicity detection*. Importantly, the robots are *oblivious* and cannot maintain state between rounds. The computation they perform is always based on the data they have collected in the *current* observation step; in the next round, they again collect the data. Such weak robots are therefore interesting from a self-stabilizing perspective: as robots do not rely on memory, an adversary cannot manipulate the memory either. Indeed, researchers have demonstrated that weak robots are sufficient to solve a wide range of problems. In particular, it has been shown that the *convergence* problem, that is, the problem of moving robots close to each other, can even be solved for very weak, oblivious robots [1, 8–10].

Further simpler robots have also been considered in the literature. They stem primarily from practical experiments with real multi-robot systems with limited cognitive capabilities [11]. The papers [11–14, 16] consider the merging assumption, i.e., two robots behave as a single robot once they become sufficiently close to each other. This assumption fares well where the scheduler is fully-synchronous or semi-synchronous because the robots wake up at the same time in the next round. In the case of an asynchronous time model, even if two robots exist at the same point at the same time, they may not necessarily have that point as the destination, or one of them may wake up while the other is idle. The robots may cross over each other. Collisions can be ignored in the case of point robots.

In this paper, we explore the *area convergence* problem, where the robots have to move inside a small area not fixed beforehand. We consider a weaker version of the robot model, named as *monoculus robots*, aptly derived from its meaning as "one-eyed". A monoculus robot contains a single camera with a viewfinder that cannot measure the distances or angles. We introduce two natural, additional capabilities, namely, *Locality Detection* and *Orthogonal Line Agreement*, for these monoculus robots. The locality detection model is motivated by, e.g., capacitive sensing or sensing differences in temperature or vibration. The orthogonal line model is practically motivated by robots having a simple compass alignment for orthogonal line agreement. The robots can move a fixed distance on each activation, which can easily be encoded in hardware as the number of rotations of the wheel or something similar.

We focus on the area convergence problem for these monoculus robots and show that the problem is already non-trivial in this setting. Not only enforcing convex hull invariants are challenging, but also the fact that observation is very limited, and we cannot detect multiplicity: As robots are also not able to perform multiplicity detection (i.e., determine how many robots are collocated at a certain point), strategies such as "move toward the center of gravity" (the direction in which most robots are located) are not possible.

1.3. Our Contributions

This paper studies area convergence problems for anonymous, autonomous, oblivious, point robots without abilities to measure distances or angles, considered as monoculus robots, under the most general asynchronous scheduling model, and makes the following contributions.

1. We study area convergence of the monoculus robots and show that these robots are incapable of achieving area convergence.
2. We present a counterexample showing that monoculus robots with the ability of measuring angles fail to converge following a naive strategy like "move along the angle bisector (of convex hull angle)".
3. We introduce Locality Detection ($\mathcal{LD}$), a notion of distance, and Orthogonal Line Agreement ($\mathcal{OLA}$), a notion of direction, accordingly.
4. We present and formally analyze deterministic and self-stabilizing distributed area convergence algorithms when monoculus robots have either $\mathcal{LD}$ or $\mathcal{OLA}$.
5. We report on the performance of our algorithms through simulation. The simulations include a comparison between both algorithms, and also show the adaptability of our algorithms to errors in look stage and move stages.
6. We show that our approach can be generalized to higher dimensions and, with a small extension, supports termination.

1.4. Related Work

The problems of gathering [1], where all the robots gather at a single point, convergence [10], where robots come very close to each other, and pattern formation [1, 3] have been studied intensively in the literature.

Flocchini et al. [7] introduced the *CORDA* or asynchronous (*ASYNC*) scheduling model for weak robots. Suzuki et al. [17] have introduced the *ATOM* or semi-synchronous (*SSYNC*) model. In [1], impossibility of gathering for $n = 2$ without assumptions on local coordinate system agreement for the *SSYNC* and the *ASYNC* model is proved. Also, for $n > 2$, it is impossible to solve gathering without assumptions on either coordinate system agreement or multiplicity detection [5]. Cohen and Peleg [9] have proposed a center of gravity algorithm for convergence of two robots in the *ASYNC* model and any number of robots in the *SSYNC* model. Flocchini et al. [18] propose an algorithm to gather robots with limited visibility and agreement in the coordinate system in the *ASYNC* model. Souissi et al. [19] have proposed an algorithm to gather robots with limited visibility with an initially unstable compass if the compass achieves stability eventually in the *SSYNC* model. For two robots with an unreliable compass, Izumi et al. [20] investigate the necessary conditions required to gather them in the *SSYNC* and the *ASYNC* model.

In the "weak robot" model, the robots can find out the location of other robots in their local coordinate system in the Look step. This, in turn, implies that a robot can measure the distance between any pair of robots, albeit in its local coordinates. It can also measure the angle between two robots, considering itself as the corner. All the algorithms exploit this location information to create an invariant point or a robot where all the other robots gather. But in this paper, we deprive the robots

Papers	Scheduling	Visibility	Merging	Distance Sensing	Angle Sensing	External Control	Movement
[1]	SSYNC	Limited	Same point	Precise	Precise	No	Precise
[11]	SSYNC	Limited	Ignored	No	Precise	No	Precise
[12]	SSYNC	Limited	Same Point	Limited	Precise	No	Precise
[13]	SSYNC	Limited	Ignored	Precise	Precise	No	Precise
[14]	SSYNC	Limited	Ignored	Precise	Precise	Yes	Precise
[15]	ASYNC	Limited	No	Precise	Precise	No	Precise
$\mathcal{LD}$	ASYNC	Unlimited	Ignored	Limited	No	No	Fixed
$\mathcal{OLA}$	ASYNC	Unlimited	Ignored	No	Relative	No	Fixed

TABLE 1. A summary of literature for robots with limited sensing

of the capability to determine the location of other robots. This leads to robots incapable of finding any kind of distance or angles.

Robots with inaccurate sensors have also been studied [10]. Previous works also study robots without the ability to measure distances but can find out the direction of the robots [11]. Subsequently, Gordon et al. also consider robots with crude distance sensing capabilities, which distinguishes the near and far robots [12]. A continuous-time model has been analyzed by Manor et al. [13] with a strategy that moves the robots towards their farthest neighbor. Manor et al. [14] show that with a common north and angle measurement capability, a swarm can converge to a disk with external guidance. These papers [11–13] assume that the robots can move precisely to a point if it lies within its visibility range. In this paper, we strip the robots of making movements to specific points, and they can only move a fixed predetermined once activated.

Note that any kind of pattern formation requires these robots to move to a particular point of the pattern. Since the monoculus robots cannot measure exact distances, they cannot stop at a particular point. Hence any kind of pattern formation algorithm described in the previous works which requires location information as input is obsolete. Gathering problem is nothing but the point formation problem [1]. Hence gathering is also not possible for the monoculus robots. A variation of gathering problem known as "Near-Gathering" has been solved for the weak robots with limited visibility, where the robots have to gather in a small disk of radius ϵ and occupy distinct locations [15]. The area convergence problem considered in this paper is similar to the near-gathering but does not have restrictions on robots occupying distinct positions.

1.5. Paper Organization

The rest of this paper is organized as follows. Section 2 introduces the necessary background, and preliminaries. Section 3 presents an impossibility result for area convergence for monoculus robots and a counterexample for monoculus robots with angle measurement capabilities following "move along angle bisector" strategy. Section 4 introduces two algorithms for area convergence with monoculus robots having either locality detection or orthogonal line agreement capability. We discuss extensions to higher dimensions and termination in Section 5. We report on simulation results in Section 6 before concluding in Section 7.

2. PRELIMINARIES

2.1. Model

We are given a system of n robots, $R = \{r_1, r_2, \cdots, r_n\}$, which are located in the Euclidean plane. We consider anonymous, autonomous, homogeneous, oblivious point robots with unlimited visibility. The robots have their local coordinate system, which may not be the same for all the robots. The robots in each round execute a sequence of *Look-Compute-Move* steps: First, each robot $r \in R$ observes other robots and obtains a set $LC = \{p_1, p_2, ..., p_k\}$. A robot is represented as p_i in r's look phase. It knows the relative order between p_i's, i.e., r knows that p_1 is towards the left of p_2 in its local view.

Second, on the basis of the observed information, it executes an algorithm which computes a direction (*Compute* step); the robot then moves in this direction (*Move* step), for a fixed distance b (the step size). The robots are silent, cannot detect multiplicity points, and can pass over each other. We ignore the collisions during movement. We denote this weaker robot model as *monoculus* robot.

We consider the most general *CORDA* or the *ASYNC* scheduling model known from weak robots [7] as well as the ATOM or Semi-Synchronous (*SSYNC*) model [17]. These models define the activation schedule of the robots: the *SSYNC* model considers instantaneous computation and movement, i.e., the robots cannot observe other robots in motion, while in the *ASYNC* model, any robot can look at any time. In the *SSYNC* model, the time is divided into global rounds, and a subset of the robots are activated in each round which finish their *Look-Compute-Move* within that round. In the case of the *ASYNC* model, there is no global notion of time. The Fully-synchronous (*FSYNC*) model is a special case of the *SSYNC* model, in which all the robots are activated in each round. The algorithms presented in this paper work in both the *ASYNC* and the *SSYNC* model. For the sake of generality, we present our proofs in terms of the *ASYNC* model.

2.2. Notation and Terminology

A *Configuration* (C) is a set containing all the robot positions in 2D. At any time t the configuration (the mapping of robots in the plane) is denoted by C_t. The convex hull of configuration C_t is denoted as CH_t. We define *Augmented Configuration* at time t (AC_t) as C_t augmented with the destinations of each robot from the most recent look state on or before t. If all the robots are idle at time t, then AC_t is the same as C_t. The convex hull of AC_t is denoted as ACH_t as shown in Fig. 1. *Area convergence* is achieved when the distance between any pair of robots is less than a predefined value ζ (and subsequently does not violate this). Our multi-robot system is vulnerable to adversarial manipulation, however, the algorithms presented in this paper are self-stabilizing [21] and robust to state manipulations. Since the robots are oblivious, they only depend on the *current state*: if the state is perturbed, the algorithms are still able to converge in a self-stabilizing manner [22].

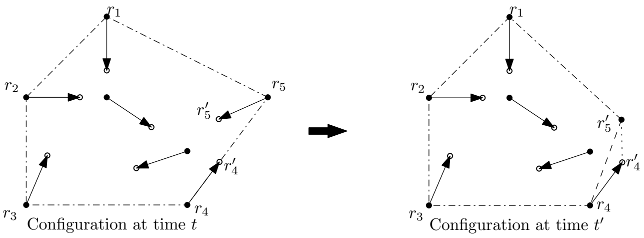

FIGURE 1. r_4' is the destination of robot r_4 from the most recent look state on or before time t, and analogously for r_5. At t, $\pentagon r_1r_2r_3r_4r_5$ is both the CH_t and ACH_t. At t' ($> t$), $\pentagon r_1r_2r_3r_4r_5'$ is $CH_{t'}$, while $\hexagon r_1r_2r_3r_4r_4'r_5'$ is $ACH_{t'}$. ACH_t contains both r_4 and r_4' because r_4 has not moved. $ACH_{t'}$ contains r_4' as a corner which is outside $CH_{t'}$, because r_5 moved to r_5'.

3. IMPOSSIBILITY

The following theorem shows that monoculus robots by themselves cannot converge deterministically. Remember that monoculus robots are incapable of measuring distances or angles. Consider a robot with a viewfinder and a mark on the camera. The robot can align the viewfinder with that robot's location, thus allowing it to move towards another robot's location. The mark specifying the positive x-axis allows the robot to keep track of one direction to perform a 360° rotation during the *look* step. The robots do not have any agreement on the mark specifying the direction of the positive x-axis.

In a deterministic algorithm, the robot cannot move along a direction that makes a particular angle with the known direction of the positive x-axis, since the robot cannot measure angles. So under the same configuration, it cannot move in the same direction that it chose previously. Thus any deterministic algorithm has to follow the direction of one of the visible robots.

3.1. Monoculus robots

Theorem 3.1. *There is no deterministic area convergence algorithm for monoculus robots without any agreement on the coordinate system in ASYNC.*

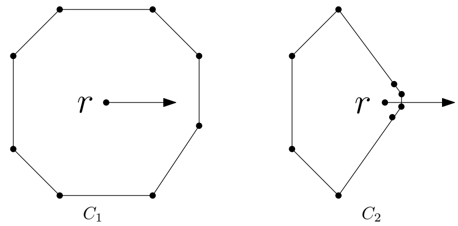

FIGURE 2. Locally indistinguishable configurations with respect to r

Proof. We prove the theorem using a symmetry argument. Consider the two configurations C_1 and C_2 in Fig. 2. In C_1, all the robots are equidistant from robot r, while in C_2, the robots are at different distances. However, the relative position of the robots is the same at r. Now considering the local view of robot r, it cannot distinguish between C_1 and C_2. Say a deterministic algorithm ϕ decides a direction of movement for robot r in configuration C_1. Since both C_1 and C_2 are the same from robot r's perspective, the deterministic algorithm outputs the same direction of movement for both cases.

Now consider the convex hull CH_1 and CH_2 of C_1 and C_2, respectively, as shown in Fig. 2. The robot r moves a distance b in one round. The distance from any point on CH_1 is more than b, but we can skew the convex hull in the direction of movement, so to make it like CH_2, where if the robot r moves a distance b, it exits CH_2. Therefore there exists a situation for any algorithm ϕ such that the area of the convex hull increases.

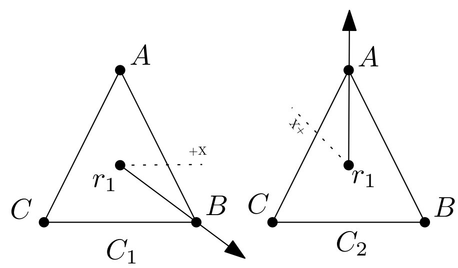

FIGURE 3. Change in local orientation of r_1

Now we have to generate a situation where the area of the convex hull increases continuously, such that the configuration diverges. Consider the configurations in Fig. 3. Suppose the robot r_1 decides to move towards B according to the configuration as shown in C_1. The adversary can always position the robot r_1 in the initial configuration such that r_1's local orientation of axes are as per C_2. Then r_1 will move towards A in C_2.

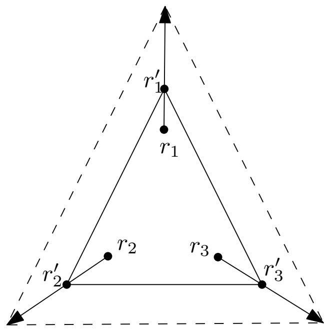

FIGURE 4. A configuration showing increase in convex hull area

To show an increase in the convex hull area, consider the configuration shown in Fig. 4. The robots r_1, r_2 and r_3 are placed by the adversary such that they move in the outward direction. The new configuration becomes analogous to the old configuration after r_1, r_2 and r_3 move. A similar situation repeats for r'_1, r'_2 and r'_3. □

3.2. Monoculus robots with angle measurement capability

Suppose the monoculus robots have the ability to measure angles. This provides another aspect of minimal enhancement. Once a robot can determine the angles of all other robots with respect to itself, it can find out whether it is on the boundary of the convex hull or not. A naïve strategy like move along the angle bisector of the convex hull angle can be considered. We now present a counterexample showing that moving along the angle bisector does not converge.

EXAMPLE 1. Consider three robots at B, C and D such that $\overline{BC} = \overline{CD}$. Let G and G' are points such that $\overline{BG}$ and $\overline{DG'}$ are parallel to the angle bisector of $\angle BCD$ as shown in Fig. 5. Suppose $\angle BCD = \theta$. The angle bisector at B of the angle $\angle CBG$ makes an angle $\theta/4$ with $\overline{BD}$. Let the angle bisector intersect $\overline{DG'}$ at B' such that $\overline{BB'} = b$, where b is the step size. Analogously, D' lies on $\overline{BG}$. So, we have $\overline{BD} = b\cos(\theta/4)$ and $\overline{B'D} = b\sin(\theta/4)$. Place another two robots at A and E which are the midpoints of $\overline{BD'}$ and $\overline{DB'}$, respectively.

Given this initial configuration at time t, the adversary activates the robots at B, C, and D at the same time. The robots at B and D move to B' and D', respectively. The adversary puts another robot at C' such that $\overline{CC'} = \overline{AB}$. With this, the new configuration at time t' has the robots at D', A, C', E and B'. The new configuration is exactly the same as the previous configuration. This configuration can be repeated infinitely many times. Note that a finite number of robots are needed on the line $\overline{CK}$ to continue this repeating configuration. This class of configurations can be constructed for any value of $\theta \leq \pi/3$. This counterexample shows that the angle bisector strategy fails to converge the robots.

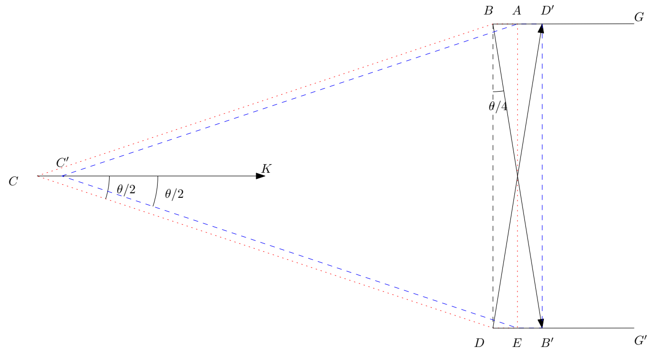

FIGURE 5. Repeating Configurations with the angle bisector strategy

4. AREA CONVERGENCE ALGORITHMS

Area convergence is the problem of moving all the robots inside a sufficiently small non-predefined area. In this section, we present area convergence algorithms for monoculus robots with additional capability where they can either have *locality detection* ($\mathcal{LD}$), a notion of distance or *orthogonal line agreement* ($\mathcal{OLA}$), a notion of direction.

Locality Detection ($\mathcal{LD}$)**:** Locality detection is the ability of a robot to determine whether its distance from any robot is greater than a predefined value c or not.

A robot with *locality detection* capability can divide the robots into two sets based on the distance from itself. So a monoculus robot with locality detection can partition the set LC to two disjoint sets LC_{local} and $LC_{non-local}$, where LC_{local} and $LC_{non-local}$ are the set of directions of robots with distances less than equal to c and more than c respectively. We assume that when a robot determines the distance of another robot in a particular direction, it can only determine the distance from the nearest robot in that direction. In other words, if there are multiple robots in a particular direction, i.e., they are collinear, then the resulting direction is part of either LC_{local} or $LC_{non-local}$ depending on the distance from the nearest robot.

Orthogonal Line Agreement ($\mathcal{OLA}$)**:** The robots agree on a pair of orthogonal lines, but can neither distinguish the two lines in a consistent way nor have a common sense of direction.

Robots with *orthogonal line agreement capability* agree on the direction of two perpendicular lines, but the lines themselves are indistinguishable: the robots neither agree on a direction (e.g., North) nor can they mark a line as, e.g., the North-South or East-West line. In other words, any two robots agreeing on the pair of orthogonal lines, either have their x-axis parallel or perpendicular to the other. In the case of parallel orientation, the plus/minus direction of the x-axis may point to the same or the opposite direction, and in the case of a perpendicular orientation, the rotation of the axis can be clockwise or counterclockwise.

In the following, we present algorithms for robots with only one of the two capabilities.

4.1. Monoculus robots with Locality Detection

In this section, we consider the area convergence problem for the monoculus robots with locality detection $\mathcal{LD}$ capability. Our claims hold for any $c \geq 2b$, where c is the predefined distance of locality detection, and b is the step size a robot moves each time it is activated. The step size b and locality detection distance c is common for all the robots. Each robot r executes Algorithm 1 (ConvergeLocality) in every time it is activated. Algorithm 1 distinguishes between two cases: (1) If the robot r only sees one other robot p, it infers that the current configuration must be a line (of 2 or more robots), and that this robot must be on the border of this line; in this case, the boundary robots always move inside (usual step size b). (2) Otherwise, a robot moves towards any visible, non-local robot (distance at least c), for a b distance (the step size). The algorithm works independent of n, the number of robots present.

Our proof unfolds in a number of lemmas followed by a theorem. First, Lemma 4.1 shows that it is impossible to have a pair of robots with a distance larger than $2c$ in the converged situation. Lemma 4.2 shows that our algorithm ensures a monotonically decreasing convex hull size. Lemma 4.3 then proves that the decrement in the perimeter for each movement is greater than a constant (the convex hull decrement is strictly monotonic). Combining all the three lemmas, we obtain the correctness proof of the algorithm. In the following, we call two robots *neighboring* if they see each other (line of sight is not obstructed by another robot).

Algorithm 1: ConvergeLocality

Input : Any arbitrary configuration C
Output: A robot p towards which r moves
if *only one robot p is visible* **then**
  Move distance b towards that robot p
else
  if $|LC_{non-local}| \geq 1$ **then**
    Move distance b towards any p, where $p \in LC_{non-local}$
  else
    Do not move `// All neighbor robots are within a distance` c

Lemma 4.1. *If there exists a pair of robots at distance more than $2c$ in a non-linear configuration, then there exists a pair of neighboring robots at distance more than c.*

Proof. Proof by contradiction. If there is a pair of robots with distance more than $2c$, then they themselves are the neighboring pair with more than c distance. To prevent them from being a neighboring pair with more than c distance, there should be at least two robots on the line joining them positioned such that each neighboring pair has a distance less than c. Since under $\mathcal{LD}$, the robots can only determine the distance of the nearest robot in a particular direction; they cannot look beyond their neighbors to find another robot at a distance of more than c. In Fig. 6, r_1 and r_4 are $2c$ apart. So r_2 and r_3 block the view such that $\overline{r_1 r_2} < c$, $\overline{r_2 r_3} < c$ and $\overline{r_3 r_4} < c$. Since it is a non-linear configuration, say robot r_5 is not on the line joining r_1 and r_4. l is the perpendicular bisector of $\overline{r_1 r_4}$. If r_5 is on the left side of l, then it is more than c distance away from r_4 and if it is on the right side of l then it is more than c distance away from r_1. If there is another robot on $\overline{r_4 r_5}$, then consider that as the new robot in a non-linear position, and we can argue similarly considering that robot to be r_5. If r_5 is on l, then $\overline{r_1 r_5} = \overline{r_4 r_5} > c$.Hence there would at least be a single robot similar to r_5 in a non-linear configuration for which the distance is more than c. □

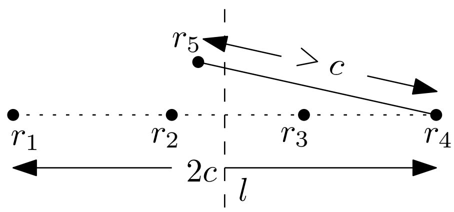

FIGURE 6. A non-linear configuration with a pair of robots at a distance $2c$

Lemma 4.2. *For any time $t' > t$ before area convergence, $ACH_{t'} \subseteq ACH_t$.*

Proof. The proof follows from a simple observation. Consider any robot r_i. If r_i decides to move towards some robot, say r_j, then it moves on the line joining two robots. There are two cases.

Case 1 : If all the robots are on a straight line, then the boundary robots move monotonically closer in each step. The distance between the end robots is a monotonically decreasing sequence until it reaches c.

Case 2 : For a non-linear configuration the robot moves when the distance between r_i and r_j is more than c and it moves only a distance b, where $c \geq 2b$. The movement path at the time when it looks, is always contained inside the CH_t, and $CH_t \subseteq ACH_t$. So the ACH_t contains its entire movement path, and it continues to do so until the robot has reached its destination. For any $t' > t$,

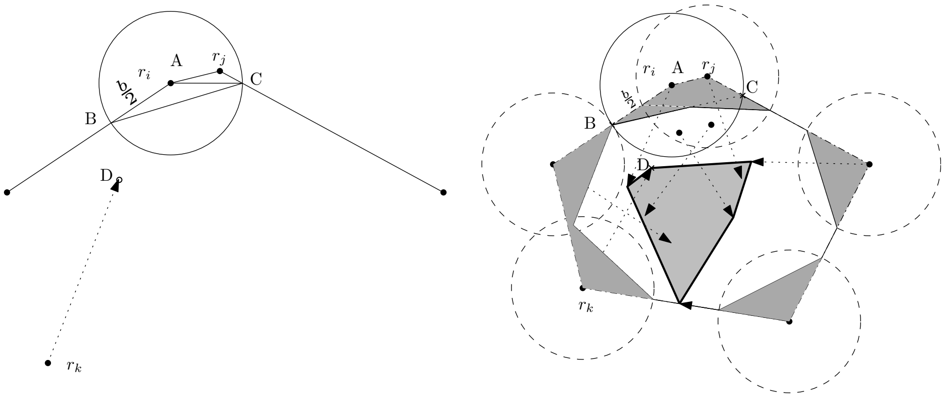

FIGURE 7. On activation r_i and r_j will move outside the solid circle inside the convex hull. The radius of the solid circle is $b/2$. The robot r_k moves a distance b towards r_i because the distance between them is more than $2b$ and stops at D. In the second figure the shadowed area is the decrement considered for each corner and the central convex hull inside solid thick lines is the new convex hull after every robot moves.

parts of the path traversed by the robots those are outside $CH_{t'}$ are removed from ACH_t (ref. the path traveled by r_5 to r_5' in Fig. 1, which is outside CH_t').

Hence $ACH_{t'} \subseteq ACH_t$. □

LEMMA 4.3. *After each robot is activated at least once, the decrement in the perimeter of the convex hull is at least* $b\left(1 - \sqrt{\frac{1}{2}\left(1 + \cos\left(\frac{2\pi}{n}\right)\right)}\right)$*, where* b *is the step size and* n *is the total number of robots.*

Proof. Suppose the n robots form a k ($k \leq n$) sided convex hull. The sum of internal angles of a k-sided convex polygon is $(k-2)\pi$. So there exists a robot r at a corner A (ref. Fig. 7) of the convex hull such that the internal angle is less than $\left(1 - \frac{2}{n}\right)\pi$, where n is the total number of robots. Let B and C be the points where the circle centered at A with radius $b/2$ intersects the convex hull. Any robot lying outside the circle will not move inside the circle according to Algorithm 1, because the maximum distance between any two points in the circle is b, and all the robots move towards a robot which is more than c distance apart and $c \geqslant 2b$. All the robots inside the circle will eventually move out once they are activated, because the robot which is activated will have to move at least b distance, and since the distance between any two points in the $b/2$ radius circle is less than or equal to b, the robot will find itself outside the $b/2$ radius circle inside the convex hull. After all the robots are activated at least once, the decrement in perimeter is at least $AB + AC - BC$. From cosine rule, $AB + AC - BC$ is

$$\frac{b}{2} + \frac{b}{2} - \sqrt{\left(\frac{b}{2}\right)^2 + \left(\frac{b}{2}\right)^2 - 2\frac{b}{2}\frac{b}{2}\cos\left(\pi - \frac{2\pi}{n}\right)}$$
$$= b\left(1 - \sqrt{\frac{1}{2}\left(1 + \cos\left(\frac{2\pi}{n}\right)\right)}\right)$$

□

REMARK 1. Let us consider a special case of the execution of the algorithm. Here all n robots are on the boundary of the convex hull with side length more than c and move only on the boundary of the convex hull. Then the n-sided polygon will again become a n-sided polygon, but the perimeter will decrease overall as a consequence of Lemma 4.3 as shown in Fig 8.

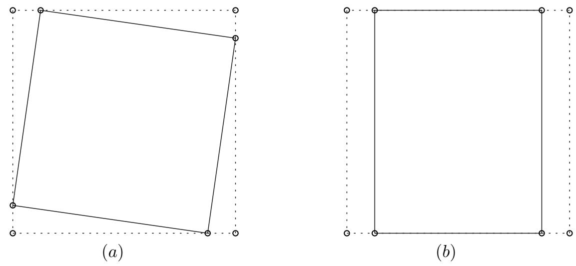

FIGURE 8. Robots on the boundary of a square are moving along the boundary (a) all in the clockwise direction, (b) not all in the same direction. The dotted line represents the configuration before movement, and the solid line represents the configuration after movement.

Since we consider the ASYNC scheduler, all robots are not activated at any time. There may be a scheduler which does not activate some robot at all. In that case, area convergence becomes impossible to achieve. To avoid this scenario, we consider a *fair scheduler*. We say a fair scheduler activates a robot infinitely many times in infinite time. This means a robot stays idle for an unpredictable but finite amount of time. So there exists a time period such that all the robots are activated at least once. We consider that time period to be a fair scheduling round. It is also known as an *epoch* in literature.

THEOREM 4.1. *(Correctness) Algorithm 1 always terminates after at most* $\Theta\left(\frac{D}{b}\right)$ *fair scheduling rounds and for any arbitrary but fixed* n*, where* D *is the diameter of smallest enclosing circle in initial* *configuration and b is the step size. After termination all the robots converge within a c radius disc.*

Proof. If a corner robot on the boundary of the convex hull is activated, then the perimeter of the convex hull decreases from Lemma 4.3. If non-corner robots are activated, then the perimeter of the convex hull remains the same. If we have a fair scheduler, the idle time for robots is unpredictable but finite. Consequently, the time between successive activations is also finite. So we can always assume a time step, which is large enough for each robot to activate at least once. The total number of robots n is finite and invariant throughout the execution, so $1-\sqrt{\frac{1}{2}\left(1+\cos\left(\frac{2\pi}{n}\right)\right)}=\delta$ is constant. Hence the decrement of perimeter is at least $b\delta$ according to Lemma 4.3. Notice that the perimeter of the convex hull is always smaller than the perimeter of the smallest enclosing circle. According to Lemma 4.1, eventually, there will not be a pair of robots with more than $2c$ distance. Note that the distance between any two points in a disc of radius c is less than or equal to $2c$. In other words, $\zeta = 2c$. Hence the robots will converge within a disc of radius c. So the perimeter of the circle at termination is $2\pi c$. Now the decrement in perimeter is $\pi D - 2\pi c$. Total time required is $\frac{\pi(D-2c)}{\delta b} = \Theta\left(\frac{D}{b}\right)$. □

4.2. Monoculus robots with Orthogonal Line Agreement

In this section we consider monoculus robots with Orthogonal Line Agreement ($\mathcal{OLA}$). Remember that monoculus robots do not agree on the coordinate system and only possess a single mark for the positive x-axis. Here we assume agreement on a pair of orthogonal lines. The angle measurement capability of the robots is relative to the lines, i.e., they can distinguish between robots located on the left or right of the orthogonal lines. Using this capability, our algorithm will distinguish between *boundary-*, *corner-* and *inner-robots*, defined in the canonical way. We note that robots can determine their type: From Fig. 9, we can observe that for r_2, all the robots lie below the horizontal line. That means, one side of the horizontal line is empty, and therefore, r_2 can figure out that it is a boundary robot. Similarly all r_i, $i \in \{2,3,4,5,6,7,8\}$ are boundary robots. Whereas, for r_1, both horizontal and vertical lines have one of the sides empty; hence r_1 is a corner robot. Other robots are all inner robots. Consequently, we define *boundary robots* as the robots having exactly one side of one of the orthogonal lines empty.

Algorithm 2 (ConvergeQuadrant) can be described as follows. A rectangle can be constructed with lines parallel to the orthogonal lines passing through boundary robots such that all the robots are inside this rectangle. In Fig. 9, each boundary robot always moves inside the rectangle perpendicular to the boundary, and the inside robots do not move. Note that the corner robot r_1 has two possible directions to move. So it moves toward any robot in that common quadrant. Gradually the distance between opposite boundaries becomes smaller and smaller, and the robots converge. In case all the robots are on a line that is parallel to either of the orthogonal lines, then the robots will find that both sides of the line are empty. In that case, they should not move. But the robots on either end of the line would only see one robot. So they would move along the line towards that robot.

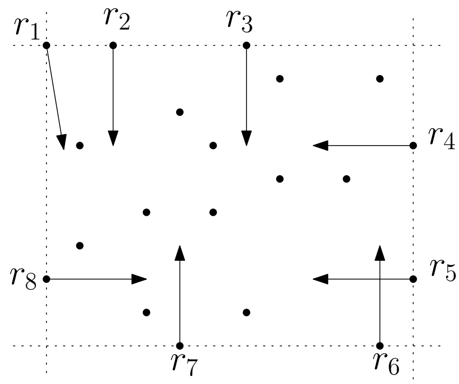

FIGURE 9. Movement direction of the boundary robots

Algorithm 2: ConvergeQuadrant

Input : Any arbitrary configuration and robot r
Output: All robots are inside a square with side $2b$

```
if only one robot is visible then
    Move towards that robot
else if r is a boundary robot then
    Move perpendicular to the boundary to the side with robots
else if r is a corner robot then
    Move towards any robot in the non-empty quadrant
else
    Do not move    // It is an inside robot
```

Theorem 4.2. *(Correctness) Algorithm 2 moves all the robots inside some $2b$-sided square in finite time, where b is the step size.*

Proof. Consider the distance between the robots on the left and right boundary. The horizontal distance between them decreases each time either of them gets activated. The rightmost robot will move towards the left, and the leftmost will move towards the right. The internal robots do not move. So in at most n activation rounds of the boundary robot, the distance between two of the boundary nodes will decrease by at least b. Hence the distance is monotonically decreasing until $2b$. Afterward, the total distance will never exceed $2b$ anymore.

If there is a corner robot present in the configuration, that robot will move towards any robot in the non-empty quadrant. So, the movement of the corner robot contributes to the decrement in the distance in both directions. If an inside robot is very close to one of the boundaries and the corner robot moves towards that robot, then the decrement in one of the dimensions can be small (an $\epsilon > 0$). Consider, for example, the configuration of a strip of width b, then the corner robot becomes the adjacent corner in the next round; this can happen only finitely many times. Each dimension converges within a distance $2b$, so in the converged state, the shape of the converged area would be $2b$-sided square, i.e., $\zeta = 2\sqrt{2}b$. □

REMARK 2. If the robots have some sense of angular knowledge, the corner robots can always move in a $\pi/4$ angle, so the decrement in both dimensions is significant, resulting in faster area convergence.

5. DISCUSSION

This section shows that our approach supports some interesting extensions.

5.1. Termination for $\mathcal{OLA}$ Model

While we only focused on area convergence and not termination so far, we can show that with a small amount of memory, termination is also possible in the $\mathcal{OLA}$ model.

Algorithm 3: CONVERGEQUADRANTTERMINATION

Input : Any arbitrary configuration and robot r with 4-bit memory
Output: All robots are inside a square with side $2b$

```
if the robot is on a boundary(ies) then
    set the corresponding bit(s) to 1
else
    Do nothing          // r is an inside robot
if r is a boundary robot and the bits corresponding to that dimension are not 1 then
    Move perpendicular to the boundary to the side with robots
else if r is a corner robot then
    if Both bits corresponding to a dimension is 1 then
        Move in other dimension to the side with robots
    else
        Move towards any robot in the non-empty quadrant
else
    Do not move  // r is not on boundary OR all four bits are 1
```

To see this, assume that each robot has a 2-bit persistent memory in the $\mathcal{OLA}$ model for each dimension, total 4-bits for two dimensions. Algorithm 2 has been modified to Algorithm 3 such that it can accommodate termination. All the bits are initially set to 0. Each robot has its local coordinate system, which remains consistent over the execution of the algorithm. The four bits correspond to four boundaries in two dimensions, i.e., left, right, top, and bottom. If a robot finds itself on one of the boundaries according to its local coordinate system, then it sets the corresponding bit of that boundary to 1. Once both bits corresponding to a dimension become 1, the robot stops moving in that dimension. Consider a robot r. Initially, it was on the left boundary in its local coordinate system. Then it sets the first bit of the pair of bits corresponding to x-axis. It moves towards the right. Once it reaches the right boundary, then it sets the second bit corresponding to x-axis to 1. Once both the bits are set to 1, it stops moving along the x-axis. Similar movement termination happens on the y-axis. Once all the 4-bits are set to 1, the robot stops moving.

5.2. Extension to d-Dimensions

Both our algorithms can easily be extended to d-dimensions. For the $\mathcal{LD}$ model, the algorithm remains exactly the same. For the proof of area convergence, similar arguments as Lemma 4.3 can be used in d dimensions. We can consider the convex hull in d-dimensions, and the boundary robots of the convex hull always move inside. The size of the convex hull reduces gradually, and the robots converge.

Analogously for the $\mathcal{OLA}$ model, the distance between two robots in the boundary of any dimension gradually decreases, and the corner robots always move inside the d-dimensional cuboid. Hence it converges. Here the robot would require $2d$ number of bits for termination.

6. SIMULATION

6.1. Simulation Setup

We now complement our formal analysis with simulations, studying the average case. We assume that initially, robots are distributed uniformly at random in a square. The step size b is equal to 1 for both the algorithms. We fix the locality detection distance $c = 2$ for Algorithm 1. Our simulations are performed in the *ASYNC* model [3]. The idle time between successive activations of each robot follows the exponential distribution with rate parameter $\lambda = 1$. We also perform simulations in the *FSYNC* model to compare convergence times of Algorithms 1 and 2 with the *ASYNC* model. We have used box plots to show the distributions. The box plots in simulation figures show four quartiles of the distribution taken from 100 executions of the algorithms. Then we connect the mean of 100 executions with line segments. In the following convergence distance and convergence time correspond to the distance and time required for area convergence.

6.2. Convergence Distance

As a baseline to evaluate performance, we consider the ratio of the actual distance traveled with optimal convergence distance. Moreover, as a lower bound for the optimal distance traveled, we consider an algorithm which converges all robots from their initial position directly to the centroid, defined as follows: $\{\bar{x}, \bar{y}\} = \left\{\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n}\right\}$ where $\{x_i, y_i\} \forall i \in \{1, 2, \cdots, n\}$ are the robots' coordinates. We calculate the distance d_i from each robot to the centroid in the initial configuration. The optimal distance we have used as convergence distance is the sum of distances from each robot to the disc of radius $b = 1$ centered at the centroid. So the total optimal convergence distance d_{opt} is given by $d_{opt} = \sum_{i=1}^{n}(d_i - 1)$, if $d_i > 1$.

In the simulation of Algorithm 1, we define d_{CL} as the cumulative number of steps taken by all the robots to converge (sometimes also known as the *work*). Now we define the performance ratio, ρ_{CL} as $\rho_{CL} = \frac{d_{CL}}{d_{opt}}$. Similarly for Algorithm 2 we define d_{CQ} and ρ_{CQ}.

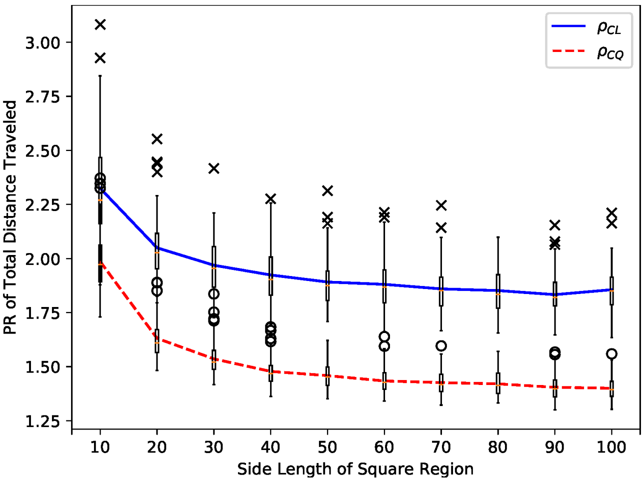

FIGURE 10. Performance Ratio of convergence distance varying size of initial deployment area with 30 robots

Figs. 10 and 11 show the comparison between the Performance Ratio (PR) for distance. We can observe that in Fig. 11, the distance traveled compared to optimal distance increases for the same size region as the number of robots increases for Algorithm 1 but it remains almost the same for Algorithm 2. We can observe that Algorithm 2 performs better than Algorithm 1. This is due to the fact that in Algorithm 2, only boundary robots move.

6.3. Convergence Time

We now compare the time of area convergence of robots. We have simulated both the algorithms in *FSYNC* and

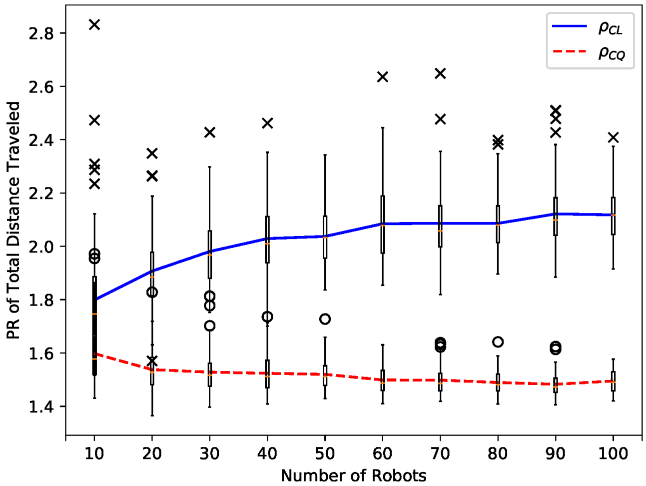

FIGURE 11. Performance Ratio of convergence distance varying number of robots with 30×30 initial deployment area

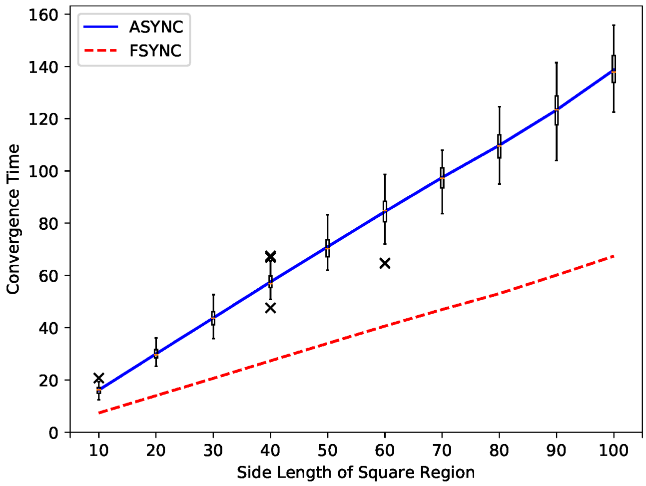

FIGURE 12. Comparison between time taken by Algorithm 1 in *FSYNC* and *ASYNC* model varying size of initial deployment area for 30 robots

ASYNC models. In Fig. 12, we plot the time taken by Algorithm 1. Similarly in Fig. 13, we plot for Algorithm 2.

We repeat the process varying the number of robots and show the results in Figs. 14 and 15. We can observe that the gap between the *FSYNC* and *ASYNC* gradually increases as the size of the area increases, but the gap remains almost constant when the number of robots increases. This is because the convergence time depends on the maximum distance needed to be traveled by a robot. If the robot is initially placed farther from the convergence area, it takes more time to reach it. But in the case of variation in the number of robots, the deployed area remains the same. Hence the robots only need to travel the same distance even if there are a lot of robots. Let us denote t_{CL} as the time taken by Algorithm 1 to achieve area convergence in the *ASYNC* model. Similarly, t_{CQ} as

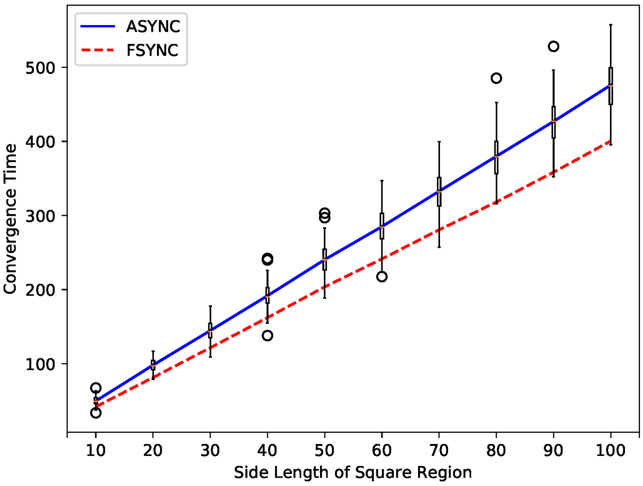

FIGURE 13. Comparison between time taken by Algorithm 2 in *FSYNC* and *ASYNC* model varying size of initial deployment area with 30 robots

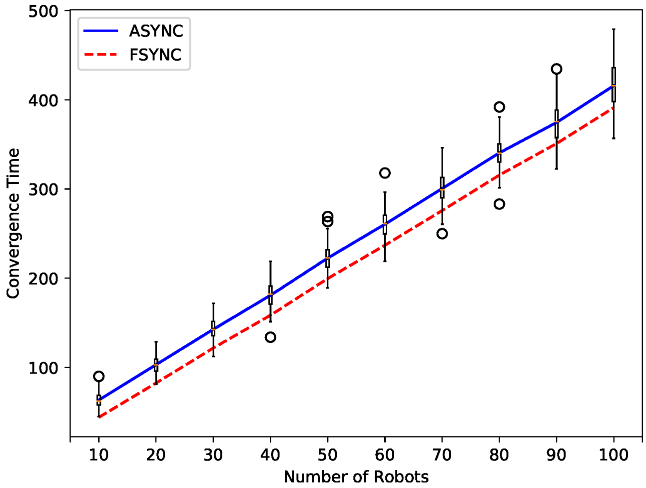

FIGURE 15. Comparison between time taken by Algorithm 2 in *FSYNC* and *ASYNC* model varying number of robots with 30×30 initial deployment area

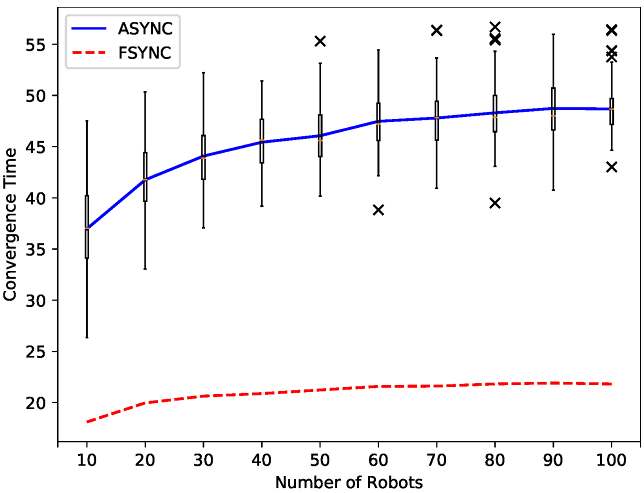

FIGURE 14. Comparison between time taken by Algorithm 1 in *FSYNC* and *ASYNC* model varying number of robots with 30×30 initial deployment area

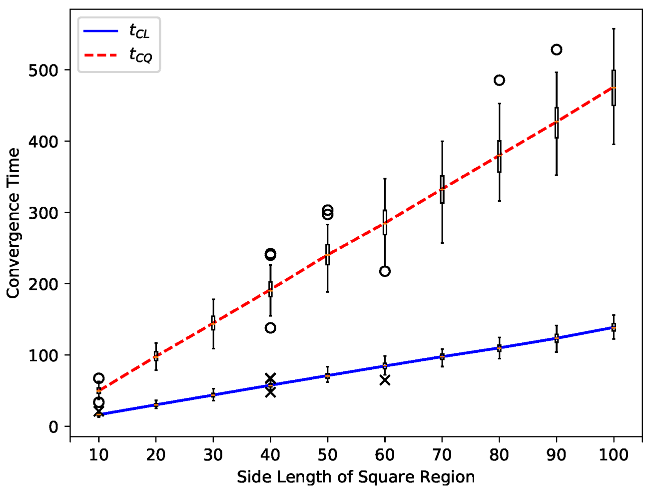

FIGURE 16. Comparison of convergence time for Algorithm 1 and 2 varying size of initial deployment area with 30 robots

the time taken by Algorithm 2. Then we compare the time taken by both the algorithms with each other in Figs. 16 and 17. We can observe that unlike the comparison of distance, Algorithm 2 takes more time to converge compared to Algorithm 1. This is because, in Algorithm 2, only boundary robots move, whereas the internal robots do not move until they become boundary robots. In the case of Algorithm 1, the internal robots also continuously move closer to the convergence area.

6.4. Sensitivity Analysis: Impact of Errors

In Figs. 18 and 19, we plot the variation of convergence time with respect to errors in moving towards a robot. A robot executing either Algorithm 1 or 2 has to move towards a robot in the move step. We consider that the direction it decides to move towards is not exactly the direction of the other robot. We term this as look error. In the plot, we execute the algorithms with look errors corresponding to a normal distribution with mean $\mu = \frac{\pi}{i}$ for $i \in \{10, 9, \cdots, 1\}$ and variance $\sigma^2 = 2\mu$. We can observe that the convergence time of Algorithm 1 remains almost the same even after the introduction of look error. In Fig. 18 and 19, we plot the convergence time varying error in moving towards a robot. Since the direction of movement of in case of Algorithm 1 is chosen at random out of all the robots at a distance more than c, moving in a different direction does not affect convergence time as shown in Fig. 18.

In the case of Algorithm 2, the error in look data may lead the robot to think that it is inside the boundary while it is actually on the boundary. We can observe that the convergence time increases as the look error increases in Fig. 19.

Similar to look error, we also introduce errors in movement. The robots are supposed to move a distance

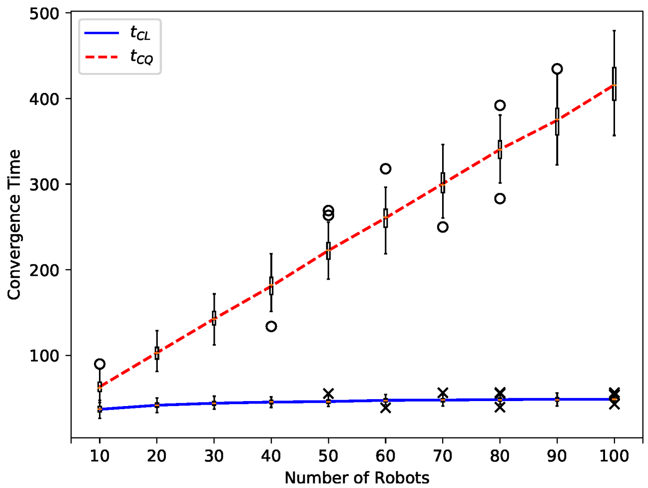

FIGURE 17. Comparison of convergence time for Algorithm 1 and 2 varying number of robots with 30×30 initial deployment area

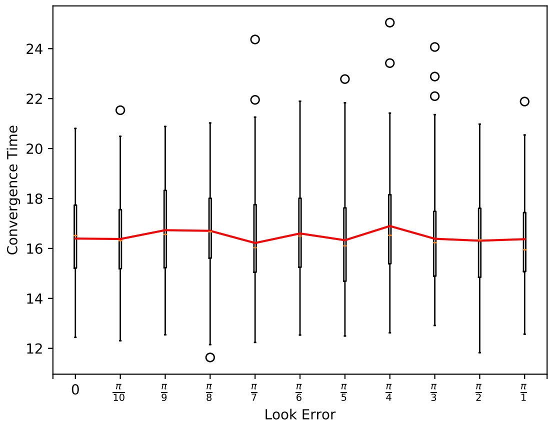

FIGURE 18. Convergence time with varying look error for Algorithm 1 with 30 robots in 10×10 initial deployment area

$b = 1$ each time they are activated. The move error makes the robots move to a distance $b + \mathcal{N}(\mu, \sigma^2)$. The move error follows a normal distribution with mean $\mu \in \{-0.5, -0.4, \cdots, 0.5\}$ and variance $\sigma^2 = |2\mu|$. We have considered the mean as both positive and negative for move error. Observe that the convergence time increases with positive error in the movement for Algorithm 1, because the robots near the convex hull may go outside of the convex hull as the error in movement increases as shown in Fig. 20. If the error in movement is negative, then the robots move less distance towards convergence area leading to an increase in convergence time. Since the robots only move inside for Algorithm 2, they converge faster as the movement distance in each step increases, as shown in Fig. 21.

Finally, we also plot convergence time with different values of c varying $\{2b, 1.9b, \cdots, b\}$. For the previous

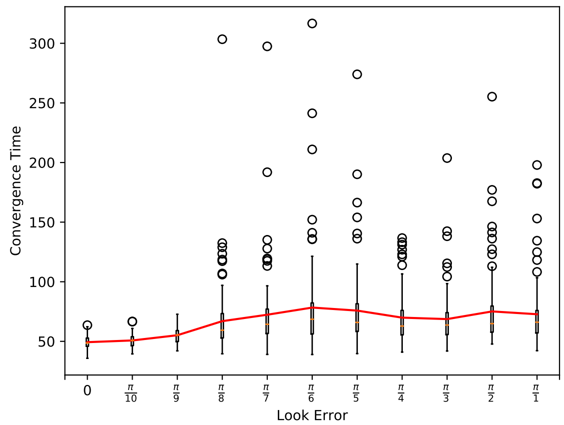

FIGURE 19. Convergence time with varying look error for Algorithm 2 with 30 robots in 10×10 initial deployment area

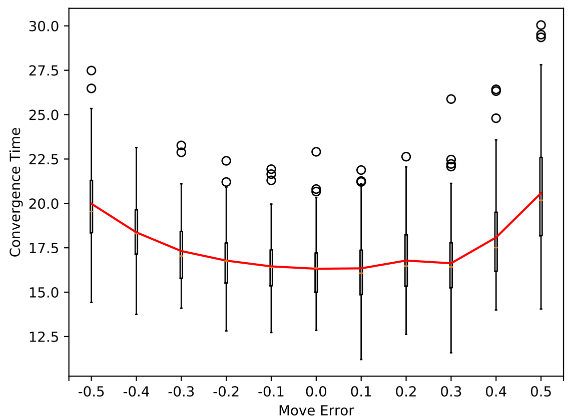

FIGURE 20. Convergence time varying move error for Algorithm 1 with 30 robots in 10×10 initial deployment area

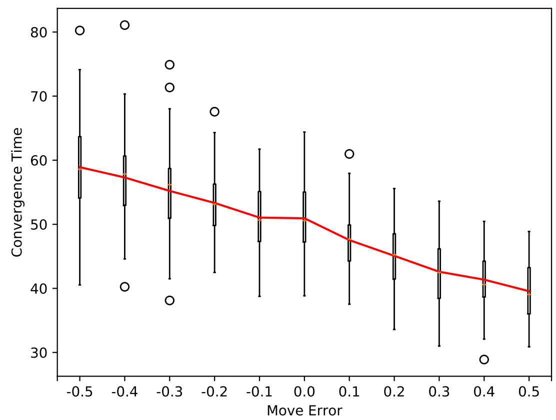

FIGURE 21. Convergence time varying move error for Algorithm 2 with 30 robots in 10×10 initial deployment area

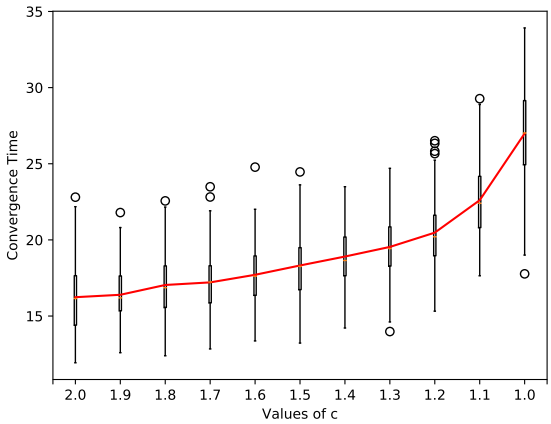

FIGURE 22. Convergence time varying the value of c in Algorithm 1 for 30 robots in 10×10 initial deployment area

simulations, we have considered the value of c to be $2b$ with $b = 1$. Fig. 22 shows that the convergence time increases as value of c approaches b. Since the decrement in the convex hull becomes gradually smaller as c approaches b, it takes more time to converge.

7. CONCLUSION

This paper considered a particularly weaker robot model known as the monoculus robot model and showed what minimum additional capabilities are required to achieve area convergence. The two approaches involve minor agreement on a common direction or a notion of distance. We presented two algorithms for area convergence with the notion of direction and distance, while showing a counterexample with respect to the notion of angles.

From simulations, we observed that the CONVERGELOCALITY algorithm converges in an almost optimal amount of time, while CONVERGEQUADRANT takes more time. But the cumulative number of steps is less for CONVERGEQUADRANT compared to CONVERGELOCALITY since only boundary robots move. In the simulations, we have also shown the adaptability and robustness of our algorithms to errors during look and move states. In particular, CONVERGELOCALITY is resilient to the error in observation and CONVERGEQUADRANT is resilient to error in movement. We believe that our work opens interesting avenues for future research. For example, it would be interesting to generalize our study to a limited visibility model.

ACKNOWLEDGEMENTS

We thank Dr. Iosif Salem for his comments regarding the counterexample. We are also grateful to the anonymous reviewers for their insightful comments. This research work is partially supported by Global Initiative for Academic Networks (GIAN), an initiative by the Government of India for Higher Education and Overseas Visiting Doctoral Scholarship (Award No. ODF/2018/001055) by the Science and Engineering Research Board (SERB), Department of Science and Technology, Government of India.

DATA AVAILABILITY

No new data were generated or analysed in support of this research.

REFERENCES

[1] Ando, H., Oasa, Y., Suzuki, I., and Yamashita, M. (1999) Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Trans. Robotics and Automation*, **15**, 818–828.

[2] Bellaiche, L. I. and Bruckstein, A. M. (2017) Continuous time gathering of agents with limited visibility and bearing-only sensing. *Swarm Intelligence*, **11**, 271–293.

[3] Flocchini, P., Prencipe, G., Santoro, N., and Widmayer, P. (2001) Gathering of asynchronous oblivious robots with limited visibility. *STACS 2001, 18th Annual Symposium on Theoretical Aspects of Computer Science, Dresden, Germany, February 15-17, 2001, Proceedings*, pp. 247–258.

[4] Fujinaga, N., Ono, H., Kijima, S., and Yamashita, M. (2010) Pattern formation through optimum matching by oblivious CORDA robots. *Principles of Distributed Systems - 14th International Conference, OPODIS 2010, Tozeur, Tunisia, December 14-17, 2010. Proceedings*, pp. 1–15.

[5] Prencipe, G. (2007) Impossibility of gathering by a set of autonomous mobile robots. *Theor. Comput. Sci.*, **384**, 222–231.

[6] Auger, C., Bouzid, Z., Courtieu, P., Tixeuil, S., and Urbain, X. (2013) Certified impossibility results for byzantine-tolerant mobile robots. *Stabilization, Safety, and Security of Distributed Systems - 15th Intl. Symposium, SSS 2013, Osaka, Japan, November 13-16, 2013. Proceedings*, pp. 178–190.

[7] Flocchini, P., Prencipe, G., Santoro, N., and Widmayer, P. (1999) Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. *Proc. ISAAC*, pp. 93–102.

[8] Cohen, R. and Peleg, D. (2004) Robot convergence via center-of-gravity algorithms. *Proc. SIROCCO*, pp. 79–88.

[9] Cohen, R. and Peleg, D. (2005) Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM J. Comput.*, **34**, 1516–1528.

[10] Cohen, R. and Peleg, D. (2008) Convergence of autonomous mobile robots with inaccurate sensors and movements. *SIAM J. Comput.*, **38**, 276–302.

[11] Gordon, N., Wagner, I. A., and Bruckstein, A. M. (2004) Gathering multiple robotic a(ge)nts with limited sensing capabilities. *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5 - 8, 2004, Proceedings*, pp. 142–153.

[12] Gordon, N., Elor, Y., and Bruckstein, A. M. (2008) Gathering multiple robotic agents with crude distance sensing capabilities. *Ant Colony Optimization and Swarm Intelligence, 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008. Proceedings*, pp. 72–83.

[13] Manor, R. and Bruckstein, A. M. (2016) Chase your farthest neighbour. *Distributed Autonomous Robotic Systems, The 13th International Symposium, DARS 2016, Natural History Museum, London, UK, November 7-9, 2016.*, pp. 103–116.

[14] Manor, R. and Bruckstein, A. M. (2018) Guidance of swarms with agents having bearing only and limited visibility sensors. *Swarm Intelligence - 11th International Conference, ANTS 2018, Rome, Italy, October 29-31, 2018, Proceedings*, pp. 44–56.

[15] Pagli, L., Prencipe, G., and Viglietta, G. (2015) Getting close without touching: near-gathering for autonomous mobile robots. *Distributed Comput.*, **28**, 333–349.

[16] Yu, J., LaValle, S. M., and Liberzon, D. (2012) Rendezvous without coordinates. *IEEE Trans. Automat. Contr.*, **57**, 421–434.

[17] Suzuki, I. and Yamashita, M. (1999) Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, **28**, 1347–1363.

[18] Flocchini, P., Prencipe, G., Santoro, N., and Widmayer, P. (2005) Gathering of asynchronous robots with limited visibility. *Theor. Comput. Sci.*, **337**, 147–168.

[19] Souissi, S., Défago, X., and Yamashita, M. (2009) Using eventually consistent compasses to gather memory-less mobile robots with limited visibility. *TAAS*, **4**, 9:1–9:27.

[20] Izumi, T., Souissi, S., Katayama, Y., Inuzuka, N., Défago, X., Wada, K., and Yamashita, M. (2012) The gathering problem for two oblivious robots with unreliable compasses. *SIAM J. Comput.*, **41**, 26–46.

[21] Dolev, S. (2000) *Self-stabilization.* MIT press.

[22] Gilbert, S., Lynch, N., Mitra, S., and Nolte, T. (2009) Self-stabilizing robot formations over unreliable networks. *ACM Trans. Auton. Adapt. Syst.*, **4**, 17:1–17:29.